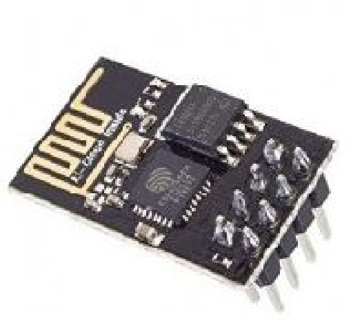
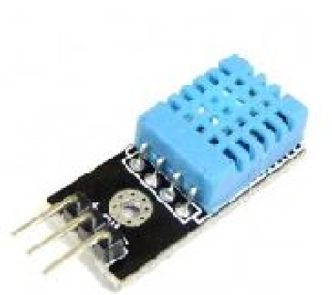
Forest Fire Surveillance Project Documentation And User Manual





Contents:

OVERVIEW

In the project we’ve worked with the arduino , WiFi module and some sensors. We’ve achieved the following goals.

- Picking up data from various sensors
- Posting data on a webpage using ESP8266 module
- Setting up a web server on ESP8266 module
- Host multiple web pages on the ESP8266 module
- Sending data from one ESP8266 module to another ESP8266 module

The purpose of this document is to help a naive user to perform all or some of these functions with ease.This document contains the following information:

- Setting up the environment for usage of the ESP8266 module
- The various connections to be made for all the above tasks.
- Screenshots of the outputs

This document contains mainly four sections:
The first section deals with gathering of data from sensors using the arduino.
The second section deals with working with the ESP8266 module.
The third deals with the communication between multiple ESP8266 modules.
The last section of this document contains the references used and other links for further help.

PICKING DATA FROM SENSORS

We have used three sensors to pick up the data:
1. Flame Sensor
2. DHT11 Humidity and Temperature Sensor
3. Smoke Sensor

All the devices have a V_{cc} and GND to provide power to the sensor and another slot through which it will send the data to the Arduino. On the Arduino side, the data collecting pin is connected to the Arduino slot. In the code for the Arduino, these analog pins are declared and the data is collected using the analogRead() function.

The connections for all three of them are as follows:

S.NO.	Arduino	Sensor
1.	V_{cc}	V_{cc}
2.	GND	GND
3.	Arduino Analog Pin	Out / Analog Out

The code for picking up data from sensors is as follows:

SMOKE SENSOR:

```
int smokeAnalogPin = A4;                          // Pin Declaration
int analogSensor = analogRead(smokeAnalogPin);    // Read Data
```

DHT11 SENSOR:

```
#include "dht.h"                                  // Required Library
int dhtAnalogPin = A0;                            // Pin Declaration
DHT.read11(dhtAnalogPin);                         // Read Data
```

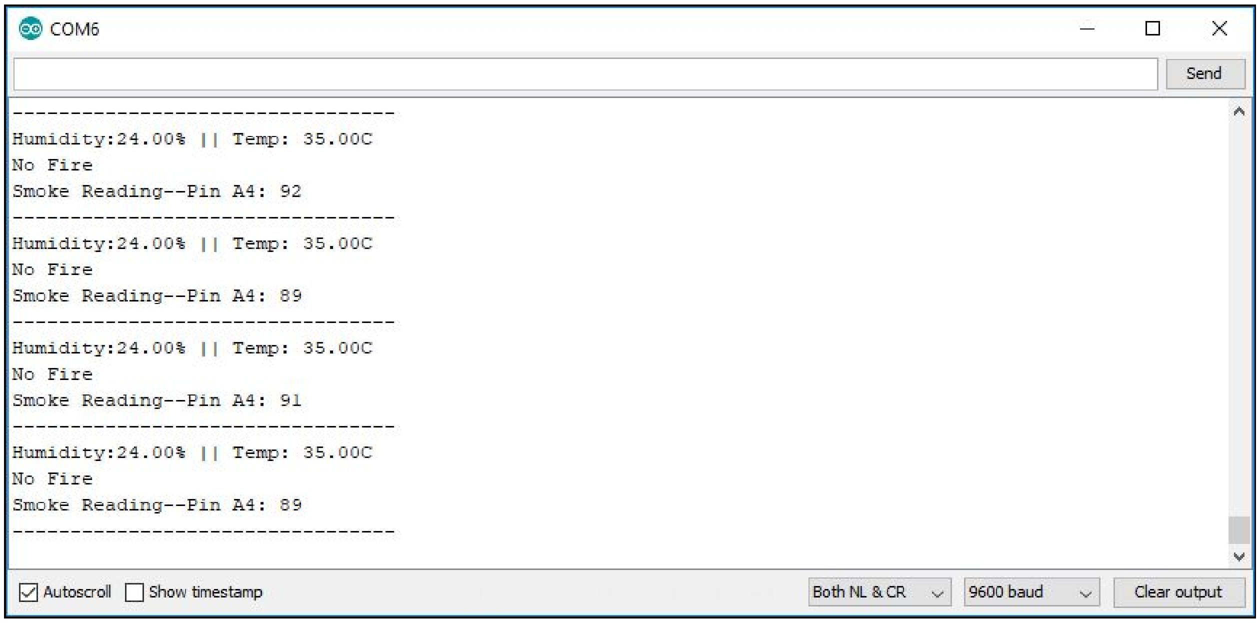
To access the readings use:

a) DHT.humidity — // Humidity Variable
b) DHT.temperature — // Temperature Variable

FLAME SENSOR:

```
int sensorReading = analogRead(A2);
int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

/* The values 0,1,2 represent the possibility and proximity of fire */
```

With this, the data for all the sensors can be collected and used in whatever way required. Sample output is as shown:



```
----------------------------------
Humidity:24.00% || Temp: 35.00C
No Fire
Smoke Reading--Pin A4: 92
----------------------------------
Humidity:24.00% || Temp: 35.00C
No Fire
Smoke Reading--Pin A4: 89
----------------------------------
Humidity:24.00% || Temp: 35.00C
No Fire
Smoke Reading--Pin A4: 91
----------------------------------
Humidity:24.00% || Temp: 35.00C
No Fire
Smoke Reading--Pin A4: 89
----------------------------------
```

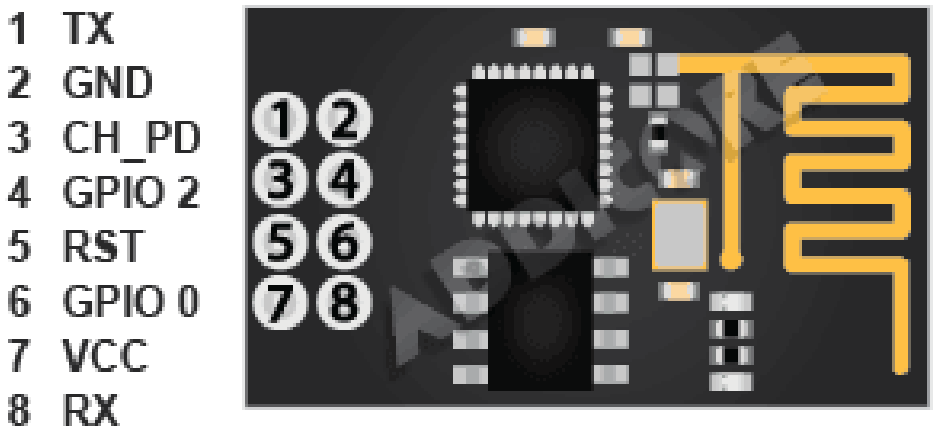
Serial Monitor Output of Sensor Data

WORKING WITH THE ESP8266 MODULE

The ESP8266 Module is a self contained system on chip (SoC) with integrated TCP/IP protocol stack.
There are mainly two ways to work with the ESP8266 module:

1. By using the preloaded firmware
2. By uploading custom code on the module

The ESP8266 module comes preloaded with AT firmware. That makes it easy to test the component and perform various operations without much effort.



Pin Diagram

Following is the circuit diagram to allow usage of the AT commands:

S.NO.	Arduino	WiFi Module
1	TX	TX
2	RX	RX
3	3.3V	V_{cc}
4	GND	GND
5	-	short CH_PD & V_{cc}

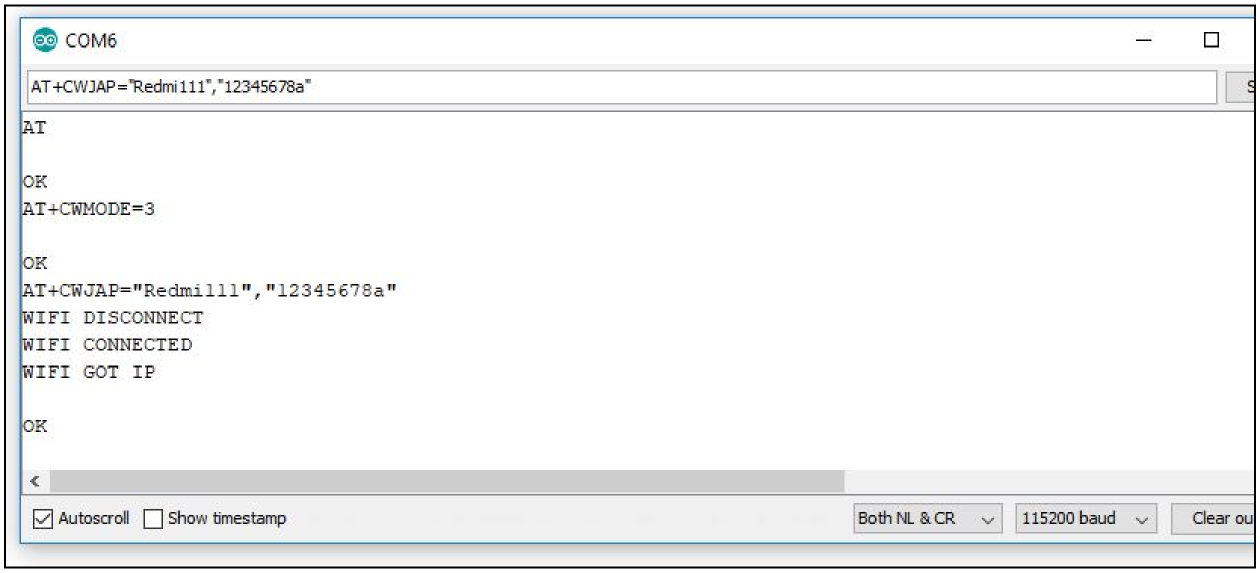
***Caution**: When using the ESP8266 in any circuit, it must be connected with the 3.3V slot. Connecting it to 5V will destroy the module.*

To use the AT firmware, follow these steps:

1. First, make the connections as shown above.
2. Open the Arduino IDE.
3. Open the serial monitor.
4. To begin, enter “AT” command to check if the setup is working.

Some other commands for the AT firmware are :

1)	AT+RST	// Reset the ESP8266 module
2)	AT	// Response check
3)	AT+CWMODE=3	// Set the module as both Station and AP
4)	AT+CIPSERVER=1,80	// Creating a server with port 80
5)	AT+CWJAP=”SSID”,”Password”	// Connect to a network



```
AT+CWJAP="Redmi111","12345678a"
```

```
AT

OK
AT+CWMODE=3

OK
AT+CWJAP="Redmi111","12345678a"
WIFI DISCONNECT
WIFI CONNECTED
WIFI GOT IP

OK
```

Connecting to WiFi with AT commands

POSTING DATA ON A WEBPAGE USING THE ESP8266 MODULE

We need to connect the WiFi module in serial with the ESP8266 module.
For the circuit , we need to put on the same connections for the ESP8266 and the sensor as above.

For uploading the data onto a webpage we can use the AT commands, we can send them using the Arduino code.
First, we have to initialize the WiFi connection:

Initiate Wifi Connection:

```
connect_wifi("AT",100);
connect_wifi("AT+CWMODE=3",100);
connect_wifi("AT+CWQAP",100);
connect_wifi("AT+RST",5000);
check4IP(5000);
if(!No_IP)
{
  connect_wifi("AT+CWJAP=\"<SSID>\",\"<PASSWORD>\"",7000);
}
get_ip();
connect_wifi("AT+CIPMUX=1",100);
connect_wifi("AT+CIPSERVER=1,80",100);
```

The “AT+CIPMUX=1” is necessary to allow multiple devices to connect to the module and view the hosted webpage.
The get_ip() function prints the IP address assigned and we can open the hosted page by typing the IP in the address bar in any device that is connected to the same network.
Now for sending the data to the webpage, we first use the following function to aggregate the data

---

Prepare HTML string:

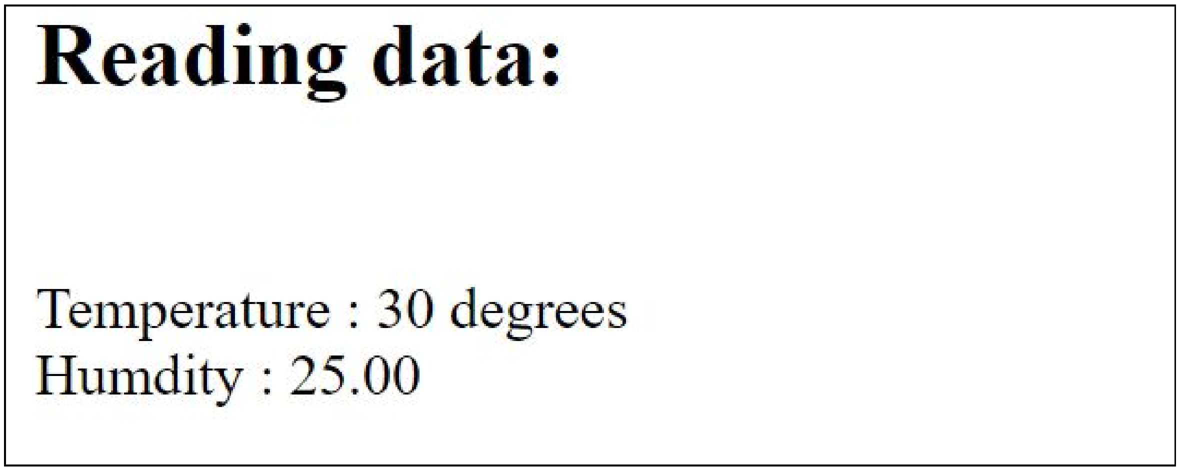
```
sendwebdata(webpage);
delay(1000);
webpage = " <h2>";
webpage+=  (String)"Humidity:"+DHT.humidity+"% || Temp: "+DHT.temperature+"C";
webpage+= "</h2>";
sendwebdata(webpage);
```

We send the created string to the webpage using the following AT command “AT+CIPSEND”.

Sending the web page:

```
    unsigned int l=webPage.length();
    Serial.print("AT+CIPSEND=0,");
    client.print("AT+CIPSEND=0,");
     Serial.println(l+2);
     client.println(l+2);
     Serial.println(webPage);
     client.println(webPage);
```

Now to send the data we can call the function in the arduino code.



Reading data:

Temperature : 30 degrees
Humdity : 25.00

Screenshot of the data posted on the webpage

UPLOADING CODE ON ESP8266

Now as mentioned in the beginning of this section , there are two ways to work with ESP8266 module. The second way i.e. uploading code in the ESP8266 module is discussed in this section.To start using the ESP8266 module in programmable mode with custom code, we first need to change some connections in the circuit. The connections to be made are as follows:

S.NO.	Arduino	ESP8266
1	TX	TX
2	RX	RX
3	3.3 V	V_{cc}
4	GND	GND
5	-	Short CH_PD & V_{cc}
6	GND	GPIO0
7	Short GND & RST	-

We then need to make some changes to the IDE settings, so that the IDE understands that the commands being sent or the code being uploaded are being done onto the WiFi module and not onto the Arduino.

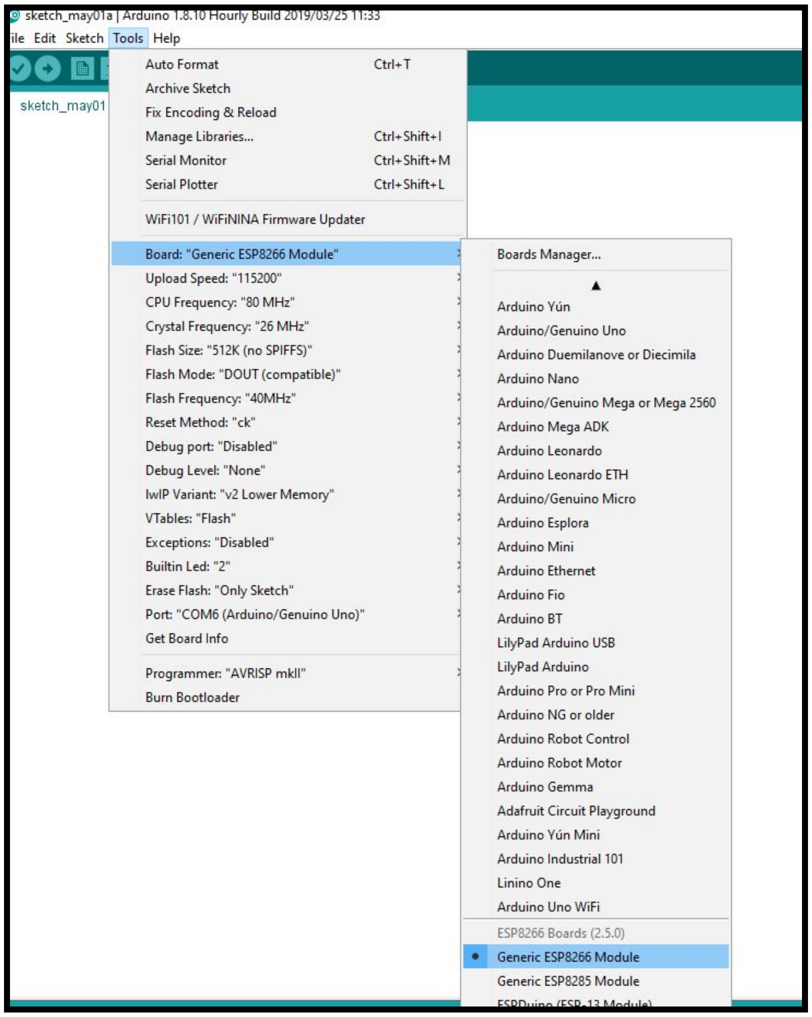
Note: Flashing/Uploading custom code onto the ESP8266 WiFi module erases the pre-existing AT firmware and unless loaded back, the AT command set doesn't work.

The circuit connections to put the ESP8266 Module into programmable mode are as follows:

To setup the environment, follow these steps:

1. Go to File→Preferences
    a) Paste the link in the “Additional Boards Manager URL” - https://arduino.esp8266.com/stable/package_esp8266com_index.json
    b) Open Tools→Board→Boards Manager.
    c) Now search for the Generic ESP8266 Module board and install.

2. Now, select Tools→Board→Generic ESP8266 module.



Board: "Generic ESP8266 Module"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Crystal Frequency: "26 MHz"
Flash Size: "512K (no SPIFFS)"
Flash Mode: "DOUT (compatible)"
Flash Frequency: "40MHz"
Reset Method: "ck"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
Exceptions: "Disabled"
Builtin Led: "2"
Erase Flash: "Only Sketch"
Port: "COM6 (Arduino/Genuino Uno)"
Programmer: "AVRISP mkII"

● Generic ESP8266 Module

Choosing ESP8266 board

Now the IDE is setup for uploading code to the ESP8266.

SETTING UP A WEB SERVER ON ESP8266 MODULE

We can host web pages on the ESP8266 module using the following libraries:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
```

First, switch on the hotspot on a device which allows multiple devices to be connected to it. This will be the device to which the WiFi modules will connect and communicate with each other. Now we will allow the ESP8266 module to connect to the hotspot network.We will enter the credentials of SSID and Password into the code of the ESP8266 module. We will now host the web pages which will basically provide data on the server.Now to access this data , we will connect to the same network and open the web pages using the ip assigned to the WiFi module.

Connects to WiFi and displays the IP:

```
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  }
Serial.println(WiFi.localIP());
```

Hosting the web pages:

There are two ways:

1. One is to send the data by hard coding it into the second argument.
2. Other is to do it via a function

Via argument:

```
  server.on("/inline", []() {
    server.send(200, "text/plain", "Hey , this is webpage created via argument");
                                  }
```

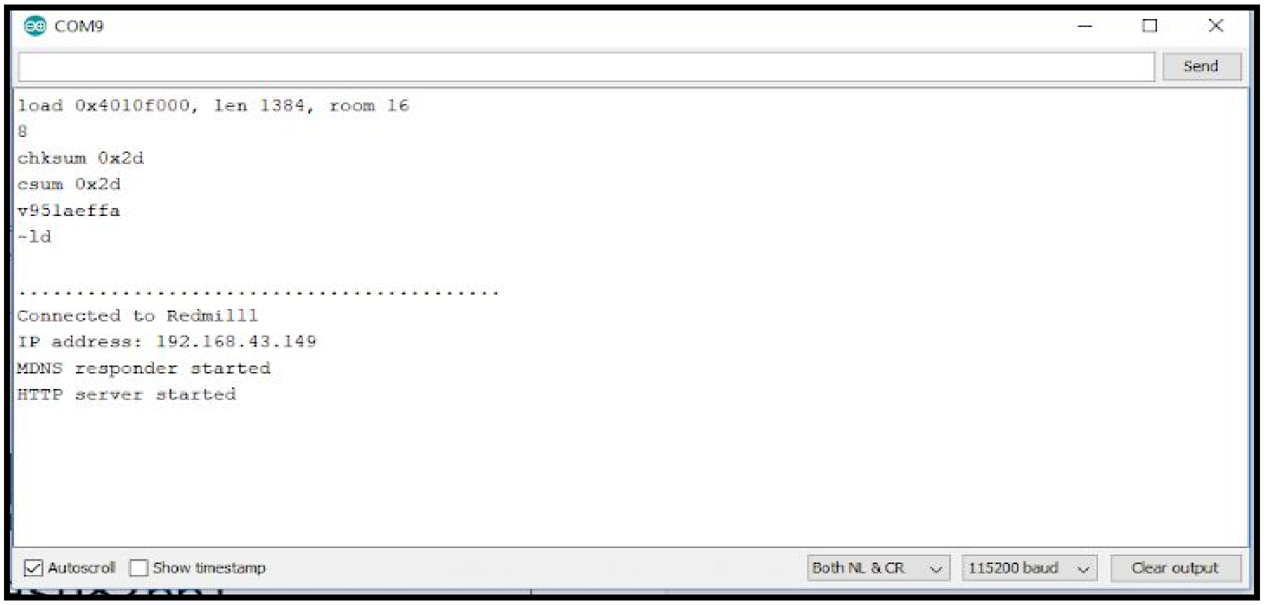
Via function:

```
 server.on("/", handleRoot);
```

```
void handleRoot() {
  server.send(200, "text/plain", "hello from esp8266!");
}
```

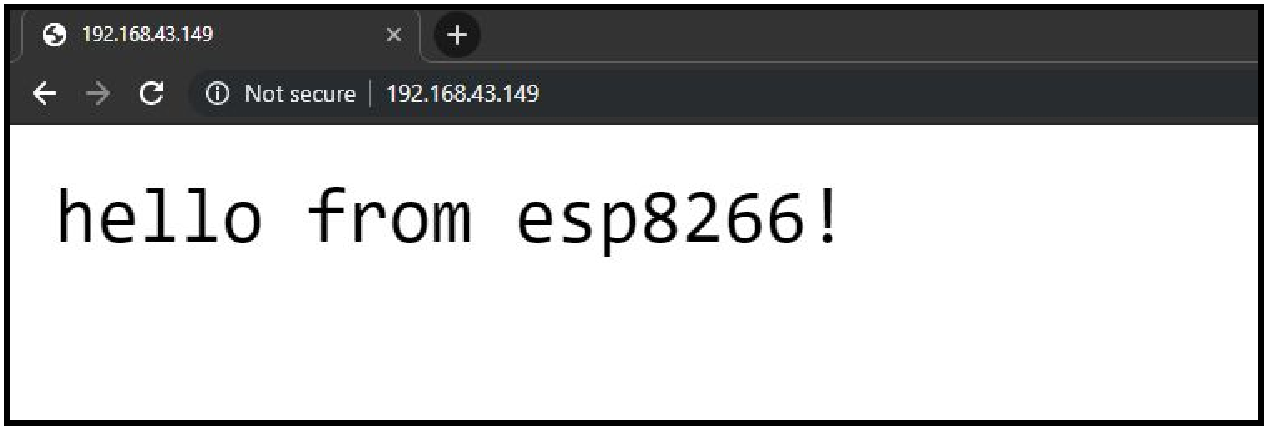
Exception Handling:

```
server.onNotFound(handleNotFound);
void handleNotFound() {
  String message = "File Not Found\n\n";

  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  server.send(404, "text/plain", message);
}
```



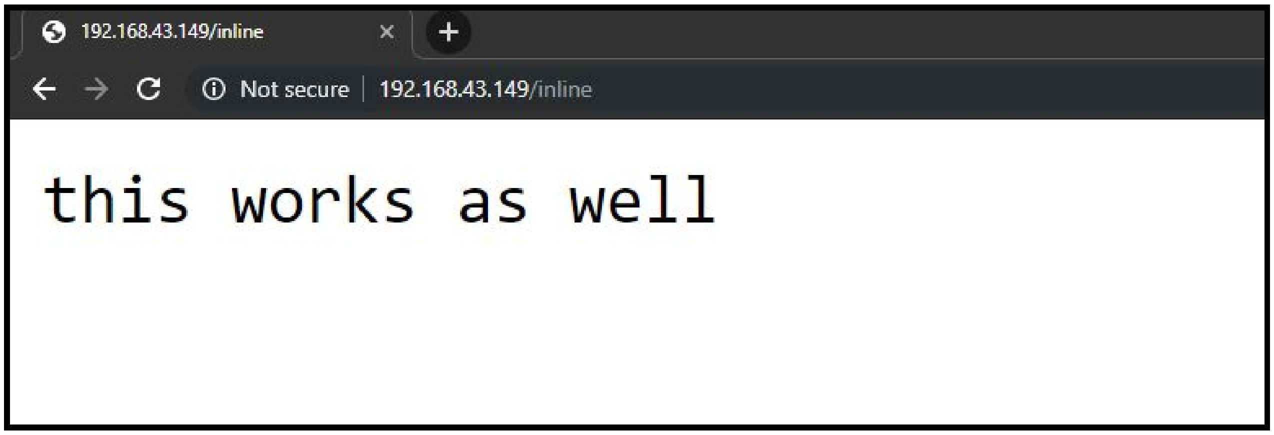
```
load 0x4010f000, len 1384, room 16
8
chksum 0x2d
csum 0x2d
v951aeffa
~ld

..........................................
Connected to Redmilll
IP address: 192.168.43.149
MDNS responder started
HTTP server started
```

Serial Monitor for ESP8266



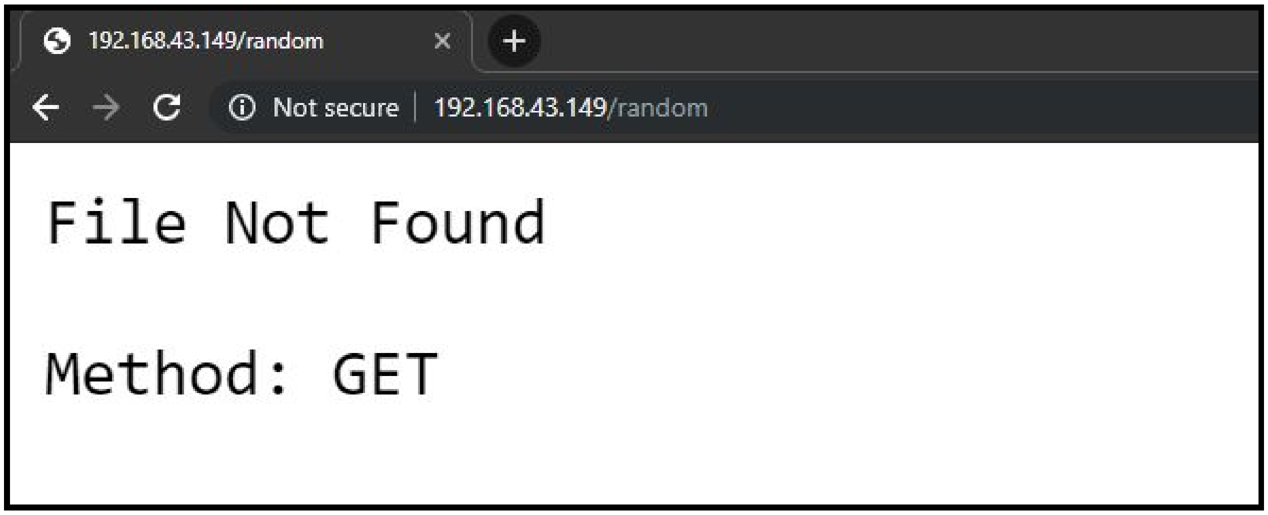
```
hello from esp8266!
```

Web page on IP of ESP8266



```
this works as well
```

Web page created with name "inline"



```
File Not Found

Method: GET
```

A “random” webpage which wasn’t actually created

SENDING DATA FROM ONE ESP8266 MODULE TO ANOTHER

Data needs to be sent from one module to another. For this we need to do the following:

1. Set one ESP8266 module to 'Access Point' mode. This makes it possible for other devices to connect to it's WiFi network.
2. We will use the second ESP8266 module to connect to the network of the first WiFi module.
3. Now, we will send the data to the hosted network via a 'GET' request.
4. Multiple devices can be connected to the first module and hence data can be sent from each and used further.

For connections , we need two ESP8266 modules with the programmable mode circuits.
Now as the first ESP8266 module receives the data through the arguments in the 'GET' request, we can utilize the data in whatever way we want.

Code for host:

Receives request:

```
void handleData(){
  Serial.println(server.arg("data"));
  server.send(200, "text/html", "Hello world");
}

  WiFi.softAP(ssid, password);
  IPAddress myIP = WiFi.softAPIP();
  server.on("/data", handleData);
  server.begin();
```

Handling the client continuously:

```
  server.handleClient();
```

Code for client:

Connect to host:

```
WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
  }
  Serial.println(WiFi.localIP());
```

Send GET request:

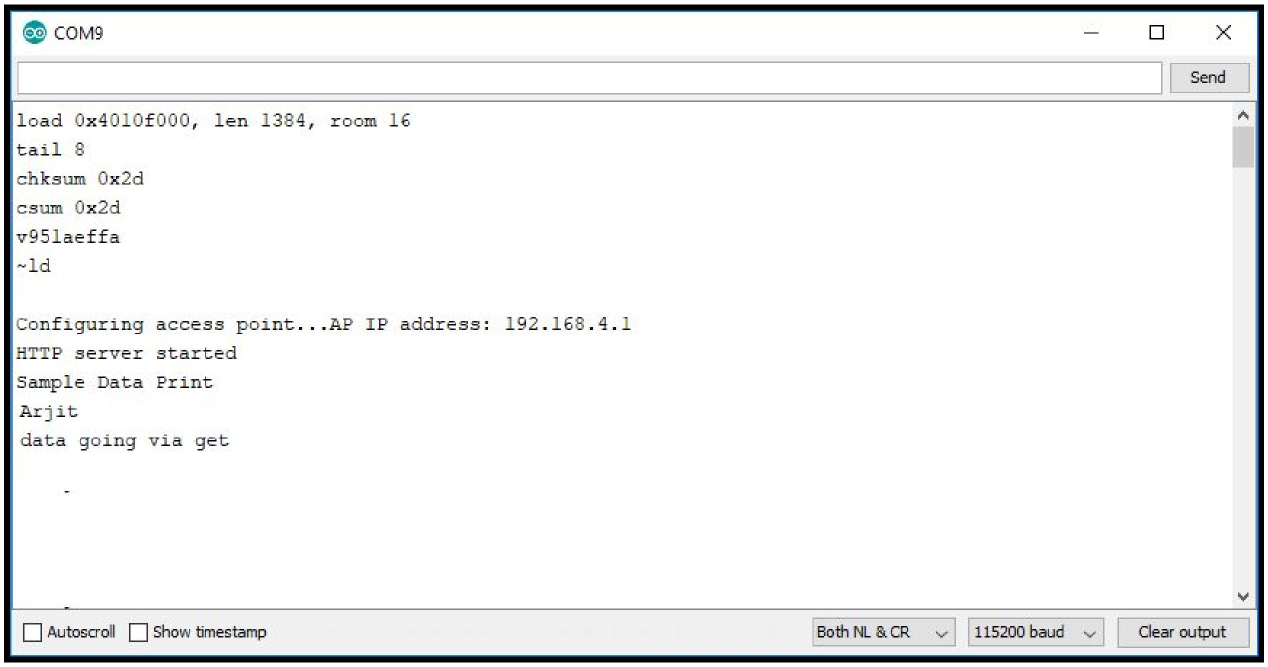
```
   HTTPClient http;
   WiFiClient client;
   String str;
   str="data=Sample%20Data%20Print";
    Serial.print("[HTTP] begin...\n");
    if (http.begin(client, "http://192.168.4.1/data?"+str)) {


      Serial.print("[HTTP] GET...\n");
      int httpCode = http.GET();
```

Successful request:

```
        Serial.printf("[HTTP] GET... code: %d\n", httpCode);
```

In case of Error:

```
        Serial.printf("[HTTP] GET... failed, error: %s\n", http.errorToString(httpCode).c_str());
        http.end();
```



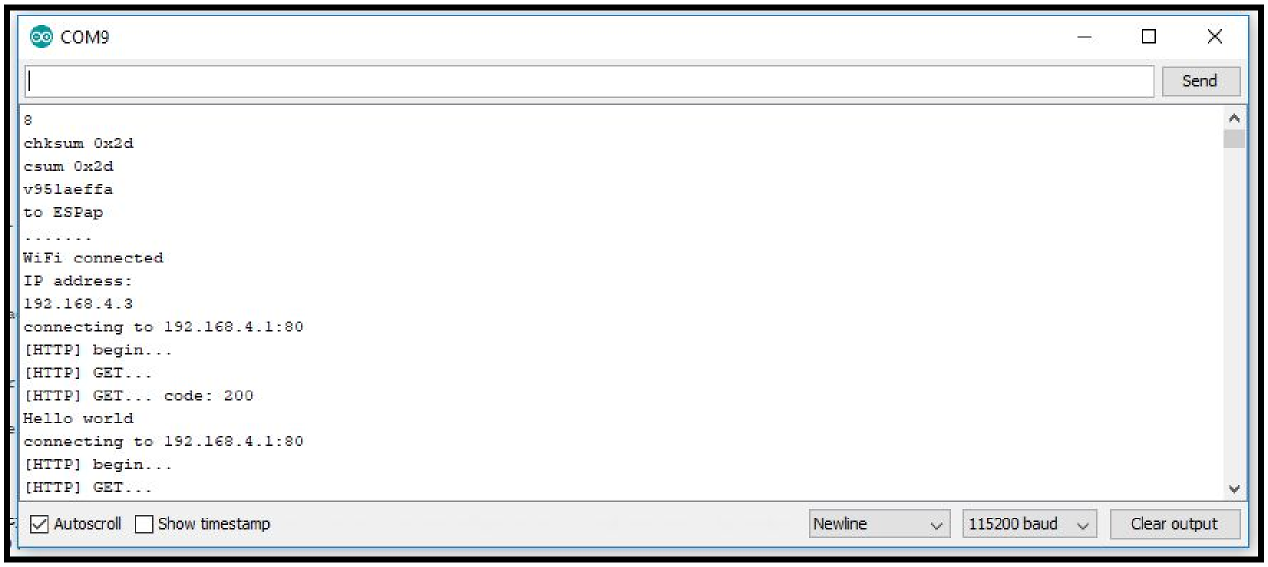
COM9

Send

```
load 0x4010f000, len 1384, room 16
tail 8
chksum 0x2d
csum 0x2d
v951aeffa
~ld

Configuring access point...AP IP address: 192.168.4.1
HTTP server started
Sample Data Print
 Arjit
 data going via get
```

Autoscroll [ ] · Show timestamp [ ] · Both NL & CR · 115200 baud · Clear output

Serial Monitor of Host



COM9

Send

```
8
chksum 0x2d
csum 0x2d
v951aeffa
to ESPap
.......
WiFi connected
IP address:
192.168.4.3
connecting to 192.168.4.1:80
[HTTP] begin...
[HTTP] GET...
[HTTP] GET... code: 200
Hello world
connecting to 192.168.4.1:80
[HTTP] begin...
[HTTP] GET...
```

Autoscroll [✓] · Show timestamp [ ] · Newline · 115200 baud · Clear output

Serial Monitor of Client

References

1. https://github.com/esp8266/Arduino
2. https://www.electronicshub.org/arduino-flame-sensor-interface/
3. http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/
4. https://randomnerdtutorials.com/guide-for-mq-2-gas-smoke-sensor-with-arduino/
5. https://www.instructables.com/id/Arduino-Esp8266-Post-Data-to-Website/
6. https://www.arduino.cc/en/main/documentation
7. https://arduino-esp8266.readthedocs.io/en/latest/